Accelerating Your Application I/O on HPC Systems

Poster Presentation

August 18, 2020

Cameron Stanavige

LLNL-PRES-813814

This work was performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under contract DE-AC52-07NA27344. Lawrence Livermore National Security, LLC

Lawrence Livermore National Laboratory

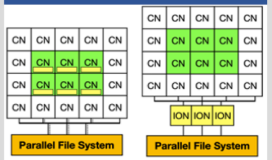
Accelerating Your Application I/O on HPC Systems

Kathryn Mohror (PI), Adam Moody, Elsa Gonsiorowski, **Cameron Stanavige**, Tony Hutter (**Lawrence Livermore National Laboratory**)
Sarp Oral (Co-PI), Feiyi Wang, Hyogi Sim, Mike Brim, Swen Boehm (**Oak Ridge National Laboratory**)
Craig Steffen, Celso Mendes (**National Center for Supercomputing Applications**)

Abstract

As the HPC community pushes towards exascale, HPC clusters are becoming increasingly complex. Part of this complexity is introduced by the addition of new storage tiers, such as burst buffers. Each tier may have different performance characteristics and may be subject to different policies regarding availability or allocation. A number of tools are under active development to improve file set management, both between tiers and within a single tier.

Types of Burst Buffers



Node Local	Machine Global
LLNL Sierra	LANL Trinity, NERSC Cori
IBM BB	Cray Datawarp
NVMe over Fabrics	IO-Nodes
Compute Node Namespace	Global Namespace
Single Allocation	Multiple Allocations

MPIFileUtils — https://github.com/HPC/mpifileutils

Suite of MPI-based file utilities to manage large datasets, including large directory trees and large files.

Tool	Speedup
dcp	51×
dcmp	60×
drm	5×

Speedup compared to regular Unix utility, using 256 MPI processes, operating on 1 million files.

Workflows

Data Migration

```
dwalk -> dcp -> dcmp --lite -> dsync
```

LC Policy Implementation

```
mpirun -np 128 drm --aggressive ./_AutoDelete
```

Available Tools
- dbcast
- dbz2
- dchmod
- dcmp
- dcp
- ddup
- dfind
- dreln
- drm
- dstripe
- dsync
- dwalk

Help wanted!
- dgre
- dparallel
- dsh
- dtar

Collaboration
- LANL
- LLNL
- ORNL
- DDN
- RedHat
- FSU
- ANU

SCR: Scalable Checkpoint Restart Library — https://github.com/LLNL/SCR

Allows MPI applications to utilize hierarchical storage for high I/O bandwidth.

Library

- Simple integration by marking application I/O phases
- Support for file-per-node I/O patterns
- Automatic management of checkpoint vs. output files

Usage

```
SCR_Start_output("dataset name", flags);
SCR_Route_file(path, newpath);

int rc = MyApp_Checkpoint(newpath);

SCR_Complete_output(rc);
```

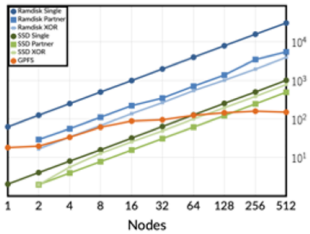
Job Management Scripts

- Reliability through cross-failure domain redundancy
- Automatic hang-detection and restart
- Scavenge data from down nodes
- Support for burst buffer asynchronous post-stage

Configurations

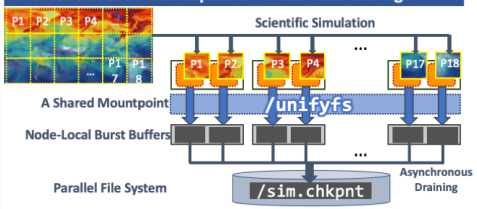
- Built-in support for various resource managers and burst buffers
- Leverage multiple asynchronous I/O technologies

SCR Scaling Performance (GB/s)



UnifyFS — https://github.com/LLNL/UnifyFS

Creates a Shared Namespace on Node-Local Storage



Using UnifyFS in a Job is Easy

```
#!/bin/bash -l

#SBATCH –N 1024
⋮

export UNIFYFS_LOGIO_SPILL_DIR=/mnt/ssd/$USER/data
unifyfs start --mount=/unifyfs --stage-in=/pfs/data

srun -n 4096 ./simulation

unifyfs terminate --stage-out=/pfs/data/out
```

Simply Change File Path

```
void checkpoint(void) {
 int rank;

 MPI_Comm_rank(MPI_COMM_WORLD, &rank);

 // file = "/pfs/shared.chpt";
 file = "/unifyfs/shared.ckpt";

 File *fs = fopen(file, "w");
 ⋮
```

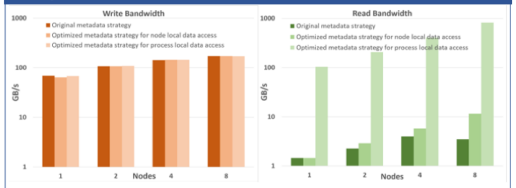
UnifyFS: Project Description and Scope

- User-level file system
- Highly-specialized for shared file access on HPC systems with distributed, node-local burst buffers
- Integration with resource managers to instantiate UnifyFS in user jobs

Will Support Common HPC I/O Use Cases

- Disjointed write and read phases, ideal for most checkpoint workload
- Scientific applications generating periodic output data
- Ensemble applications sharing data through files
- Will support HPC I/O libraries such as HDF5, ADIOS, MPI-I/O, PnetCDF

UnifyFS Performance on Summit



- Results on Summit show scalable write performance for UnifyFS with shared files on burst buffers
- Recent optimizations** can speed up reads by up to 236x over our baseline performance

** in the case where a process only reads in the bytes it wrote

This work was performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under Contract DE-AC52-07NA27344..
This research was supported by the Exascale Computing Project (ECP), Project Number: 17-SC-20-SC, a collaborative effort of two DOE organizations - the Office of Science and the National Nuclear Security Administration, responsible for the planning and preparation of a capable exascale ecosystem, including software, applications, hardware, advanced system engineering and early testbed platforms, to support the nation's exascale computing imperative.

Lawrence Livermore National Laboratory
OAK RIDGE National Laboratory
NCSA

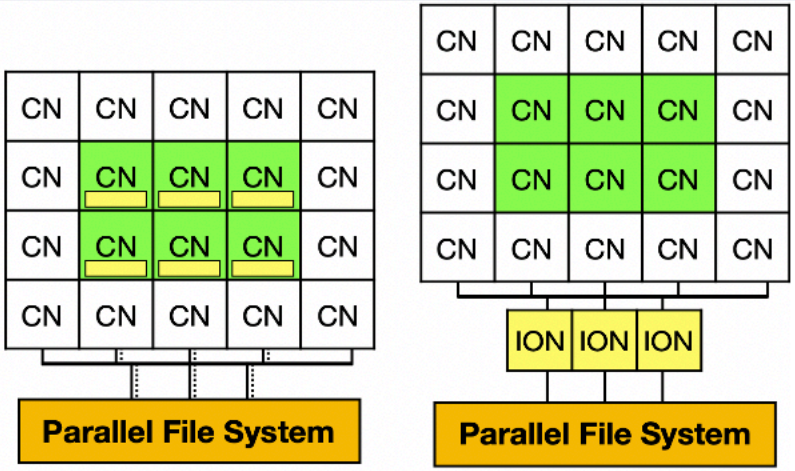
Accelerating Your Application I/O on HPC Systems

Kathryn Mohror (PI), Adam Moody, Elsa Gonsiorowski, **Cameron Stanavige**, Tony Hutter (**Lawrence Livermore National Laboratory**)
Sarp Oral (Co-PI), Feiyi Wang, Hyogi Sim, Mike Brim, Swen Boehm (**Oak Ridge National Laboratory**)
Craig Steffen, Celso Mendes (**National Center for Supercomputing Applications**)

Abstract

As the HPC community pushes towards exascale, HPC clusters are becoming increasingly complex. Part of this complexity is introduced by the addition of new storage tiers, such as burst buffers. Each tier may have different performance characteristics and may be subject to different policies regarding availability or allocation. A number of tools are under active development to improve file set management, both between tiers and within a single tier.

Types of Burst Buffers



Node Local	Machine Global
LLNL Sierra	LANL Trinity, NERSC Cori
IBM BB	Cray Datawarp
NVMe over Fabrics	IO-Nodes
Compute Node Namespace	Global Namespace
Single Allocation	Multiple Allocations

MPIFileUtils

https://github.com/HPC/mpifileutils

Suite of MPI-based file utilities to manage large datasets, including large directory trees and large files.

Tool	Speedup
dcp	51×
dcmp	60×
drm	5×

Speedup compared to regular Unix utility, using 256 MPI processes, operating on 1 million files.

Workflows

Data Migration

```
dwalk -> dcp -> dcmp --lite -> dsync
```

LC Policy Implementation

```
mpirun -np 128 drm --aggressive ./_AutoDelete
```

Available Tools

- dbcast
- dbz2
- dchmod
- dcmp
- dcp
- ddup
- dfind
- dreln
- drm
- dstripe
- dsync
- dwalk

Help wanted!

- dgre
- dparallel
- dsh
- dtar

Collaboration

- LANL
- LLNL
- ORNL
- DDN
- RedHat
- FSU
- ANU

Suite of MPI-based file utilities to manage large datasets, including large directory trees and large files.

Tool	Speedup
dcp	51×
dcmp	60×
drm	5×

Speedup compared to regular Unix utility, using 256 MPI processes, operating on 1 million files.

Workflows

Data Migration

```
dwalk -> dcp -> dcmp --lite -> dsync
```

LC Policy Implementation

```
mpirun -np 128 drm --aggressive ./_AutoDelete
```

Available Tools

- dbcast
- dbz2
- dchmod
- dcmp
- dcp
- ddup
- dfind
- dreln
- drm
- dstripe
- dsync
- dwalk

Help wanted!

- dgre
- dparallel
- dsh
- dtar

Collaboration

- LANL
- LLNL
- ORNL
- DDN
- RedHat
- FSU
- ANU

SCR: Scalable Checkpoint Restart Library

https://github.com/LLNL/SCR

Allows MPI applications to utilize hierarchical storage for high I/0 bandwidth.

Library

- Simple integration by marking application I/O phases
- Support for file-per-node I/O patterns
- Automatic management of checkpoint vs. output files

Usage

```
SCR_Start_output("dataset name", flags);
SCR_Route_file(path, newpath);

int rc = MyApp_Checkpoint(newpath);

SCR_Complete_output(rc);
```

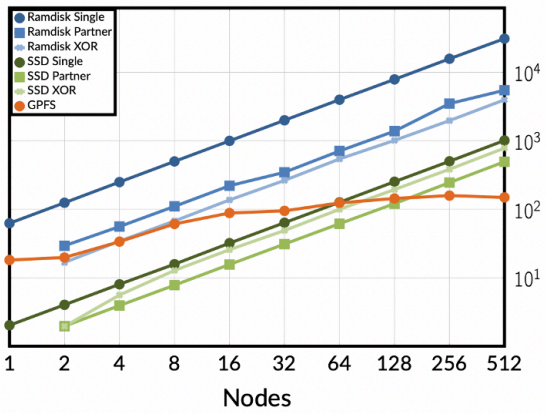
Job Management Scripts

- Reliability through cross-failure domain redundancy
- Automatic hang-detection and restart
- Scavenge data from down nodes
- Support for burst buffer asynchronous post-stage

Configurations

- Built-in support for various resource managers and burst buffers
- Leverage multiple asynchronous I/O technologies

SCR Scaling Performance (GB/s)



Library

- Simple integration by marking application I/O phases
- Support for file-per-node I/O patterns
- Automatic management of checkpoint vs. output files

Usage

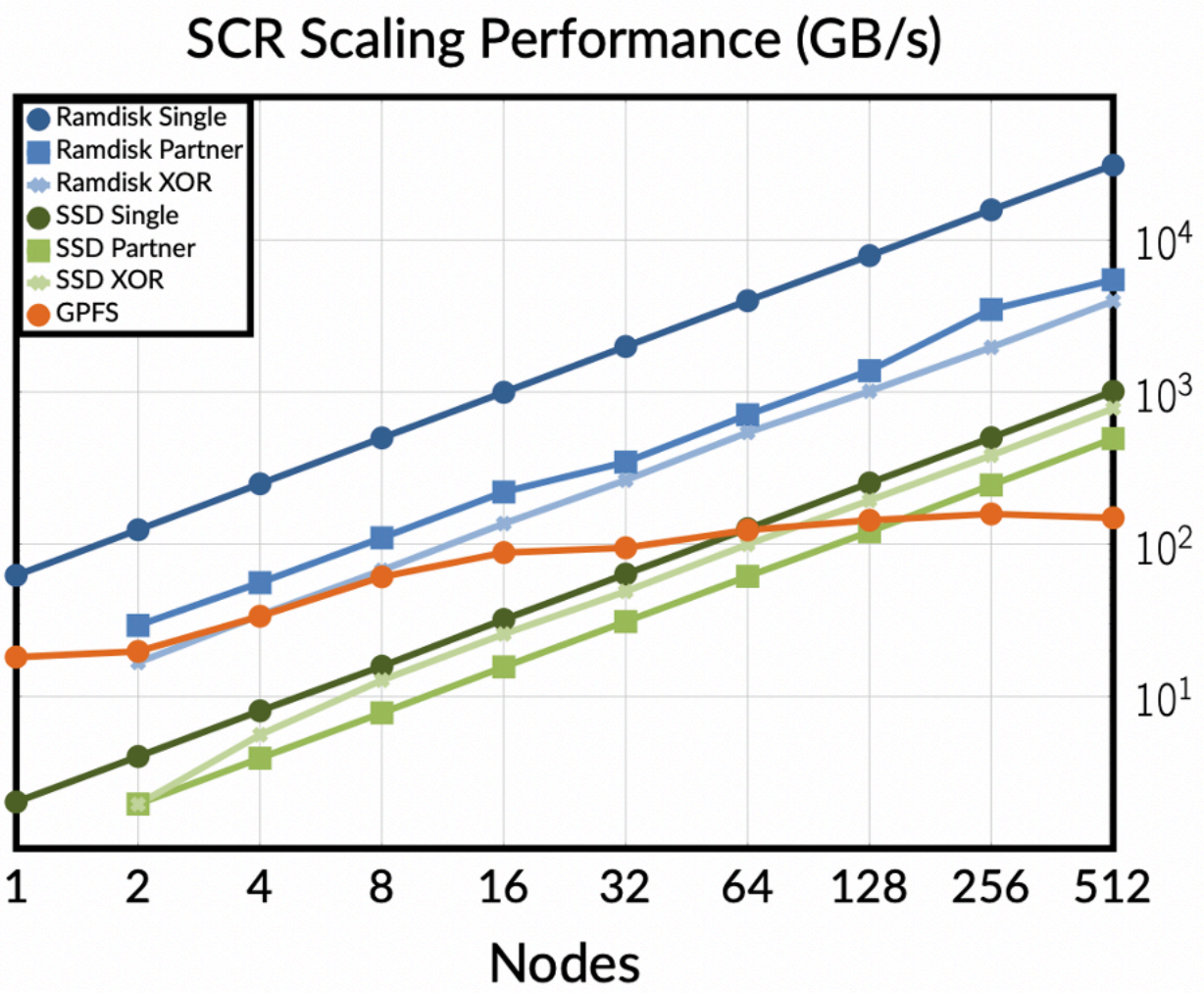
```
SCR_Start_output(”dataset name”, flags);
SCR_Route_file(path, newpath);

int rc = MyApp_Checkpoint(newpath);

SCR_Complete_output(rc);
```

Job Management Scripts

- Reliability through cross-failure domain redundancy
- Automatic hang-detection and restart
- Scavenge data from down nodes
- Support for burst buffer asynchronous post-stage

Configurations

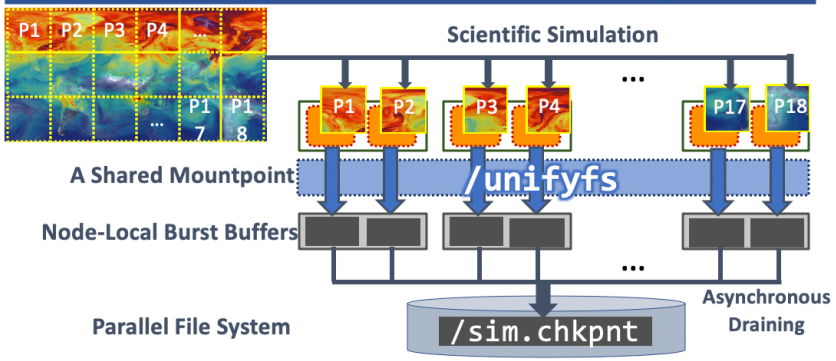
- Built-in support for various resource managers and burst buffers
- Leverage multiple asynchronous I/O technologies


SCR Scaling Performance (GB/s)
Ramdisk Single
Ramdisk Partner
Ramdisk XOR
SSD Single
SSD Partner
SSD XOR
GPFS
10^4
10^3
10^2
10^1
1
2
4
8
16
32
64
128
256
512
Nodes

UnifyFS

https://github.com/LLNL/UnifyFS

Creates a Shared Namespace on Node-Local Storage



Using UnifyFS in a Job is Easy

```
#!/bin/bash -l

#SBATCH -N 1024
⋮

export UNIFYFS_LOGIO_SPILL_DIR=/mnt/ssd/$USER/data
unifyfs start --mount=/unifyfs --stage-in=/pfs/data

srun -n 4096 ./simulation

unifyfs terminate --stage-out=/pfs/data/out
```

Simply Change File Path

```
void checkpoint(void) {
 int rank;

 MPI_Comm_rank(MPI_COMM_WORLD, &rank);

 // file = "/pfs/shared.chpt";
 file = "/unifyfs/shared.ckpt";

 File *fs = fopen(file, "w");
 ⋮
```

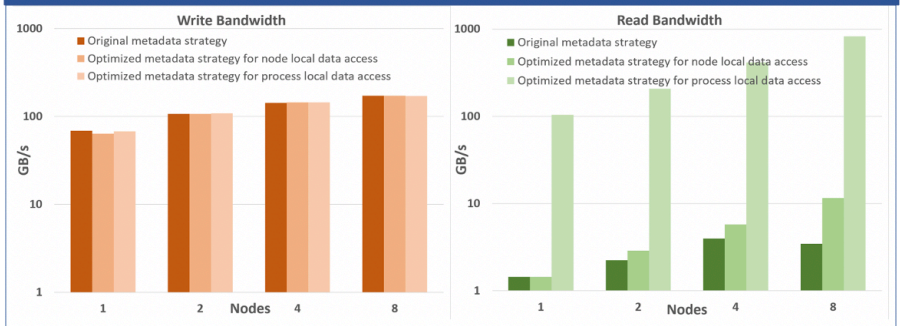
UnifyFS: Project Description and Scope

- User-level file system
- Highly-specialized for shared file access on HPC systems with distributed, node-local burst buffers
- Integration with resource managers to instantiate UnifyFS in user jobs

Will Support Common HPC I/O Use Cases

- Disjointed write and read phases, ideal for most checkpoint workload
- Scientific applications generating periodic output data
- Ensemble applications sharing data through files
- Will support HPC I/O libraries such as HDF5, ADIOS, MPI-I/O, PnetCDF

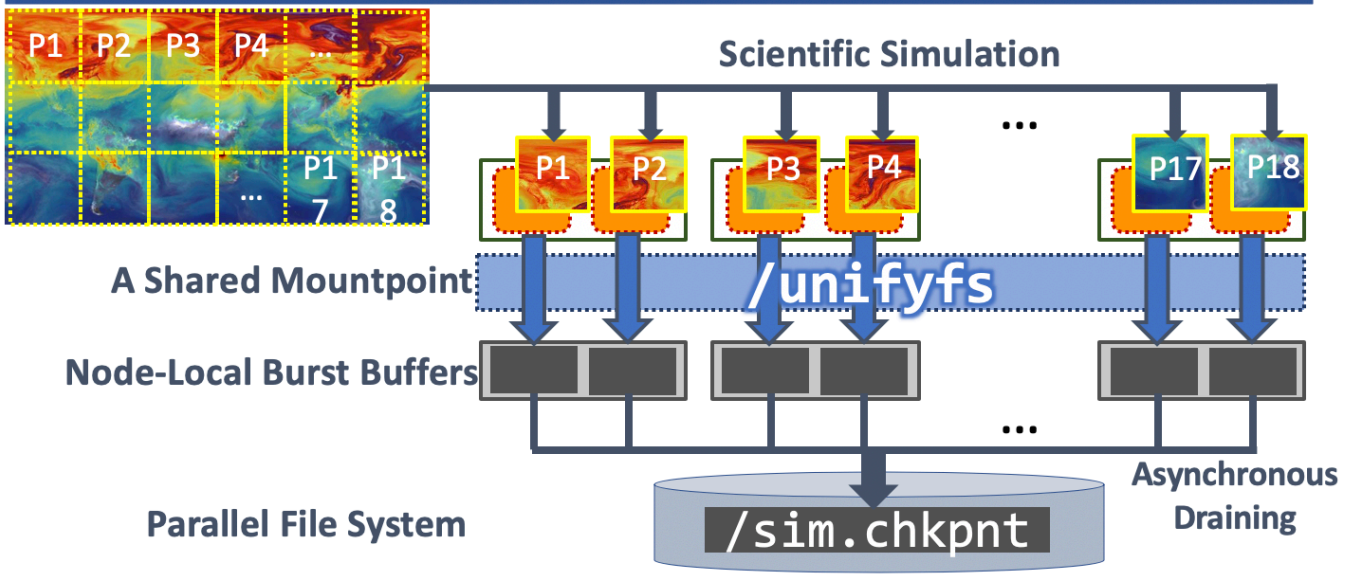
UnifyFS Performance on Summit



- Results on Summit show scalable write performance for UnifyFS with shared files on burst buffers
- Recent optimizations** can speed up reads by up to 236x over our baseline performance

** in the case where a process only reads in the bytes it wrote

Creates a Shared Namespace on Node-Local Storage



UnifyFS: Project Description and Scope

- User-level file system
- Highly-specialized for shared file access on HPC systems with distributed, node-local burst buffers
- Integration with resource managers to instantiate UnifyFS in user jobs

Will Support Common HPC I/O Use Cases

- Disjointed write and read phases, ideal for most checkpoint workload
- Scientific applications generating periodic output data
- Ensemble applications sharing data through files
- Will support HPC I/O libraries such as HDF5, ADIOS, MPI-I/O, PnetCDF

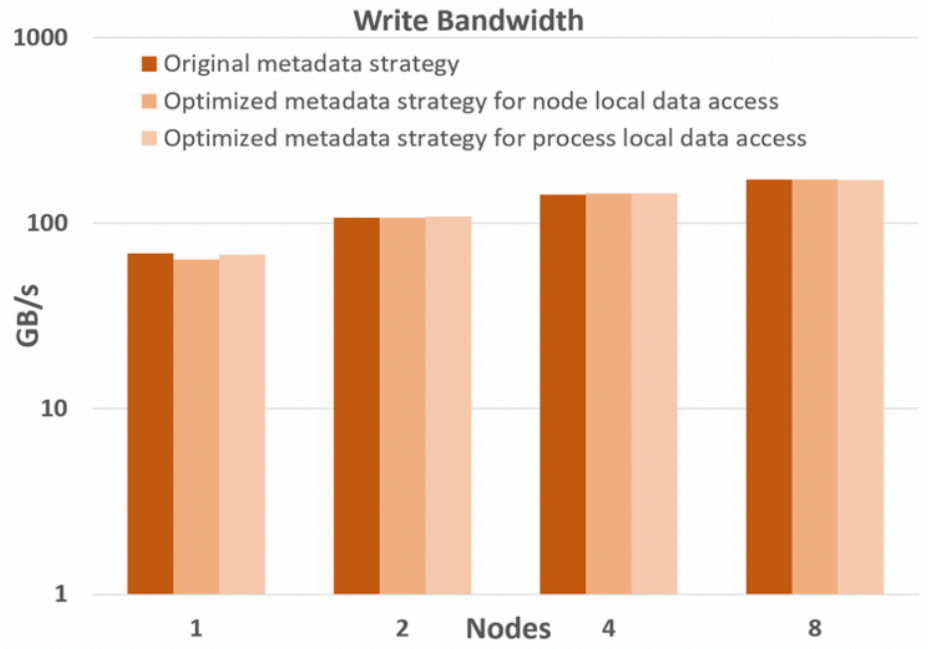
Using UnifyFS in a Job is Easy

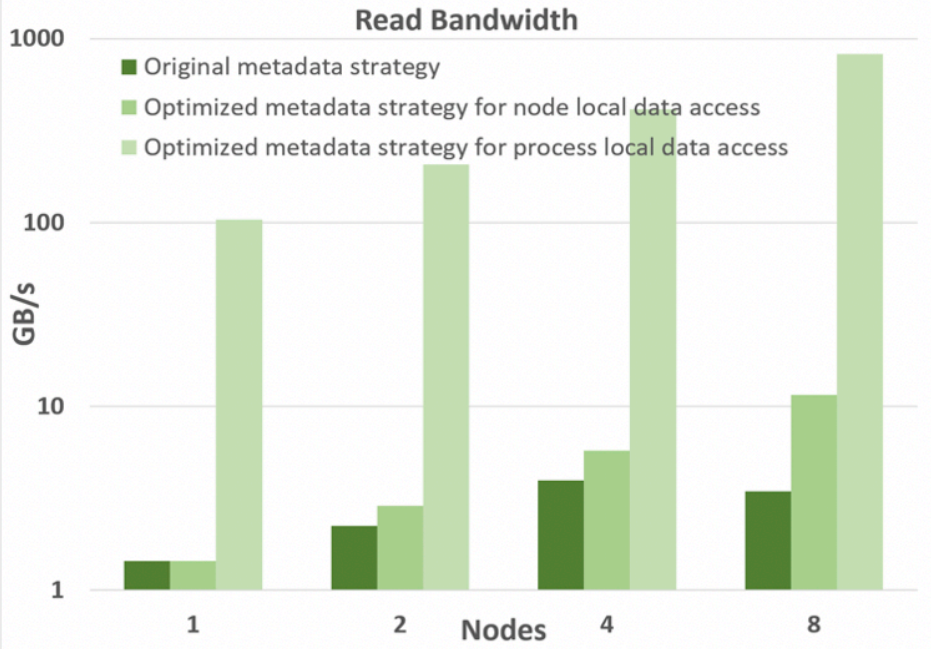
```
#!/bin/bash -l

#SBATCH –N 1024
⋮

export UNIFYFS_LOGIO_SPILL_DIR=/mnt/ssd/$USER/data
unifyfs start --mount=/unifyfs --stage-in=/pfs/data

srun –n 4096 ./simulation

unifyfs terminate --stage-out=/pfs/data/out
```

Simply Change File Path

```
void checkpoint(void) {
 int rank;

 MPI_Comm_rank(MPI_COMM_WORLD, &rank);

 // file = “/pfs/shared.chpt”;
 file = “/unifyfs/shared.ckpt”;

 File *fs = fopen(file, “w”);
 ⋮
```

UnifyFS Performance on Summit





- Results on Summit show scalable write performance for UnifyFS with shared files on burst buffers
- Recent optimizations** can speed up reads by up to 236x over our baseline performance

** in the case where a process only reads in the bytes it wrote

This work was performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under Contract DE-AC52-07NA27344..
This research was supported by the Exascale Computing Project (ECP), Project Number: 17-SC-20-SC, a collaborative effort of two DOE organizations - the Office of Science and the National Nuclear Security Administration, responsible for the planning and preparation of a capable exascale ecosystem, including software, applications, hardware, advanced system engineering and early testbed platforms, to support the nation's exascale computing imperative.

Lawrence Livermore National Laboratory

OAK RIDGE National Laboratory

NCSA

https://github.com/HPC/mpifileutils

https://github.com/LLNL/SCR

https://github.com/LLNL/UnifyFS